|  |
| --- |
| [Forfatterens navn]  19-09-2022 |

|  |
| --- |
| [Firmanavn] |
| Processrapport |
| SmartWeight |



|  |
| --- |
| **Elev:**  Daniel Simonsen |
| **Firma:**  Skolepraktik |
| **Projekt:**  SmartWeight |
| **Uddannelse:**  Datateknikker m. speciale i programmering |
| **Projektperiode:**  19/09/2022 – 23/11/2022 |
| **Afleveringsdato:**  23/11/2022 |
| **Fremlæggelsesdato:**  24/11/2022 |
| **Vejledere:**  Navn på vejledere |
|  |

# Titelblad



Techcollege Aalborg,

Struervej 70,

9220 Aalborg

# (Indeks)

Afgræsning; 1

Diskussion; 2

Indledning; 1

# Læsevejledning

I læsevejledningen beskriver man hvordan rapporterne anbefales at læses og evt. andre skriftlige formalia, som forkortelser og engelske fagord.

*Alle overskrifter i parentes er ikke et krav at have med.  
Nogen overskrifter skal omskrives for at passe ind.*

# Forord

Hvad skal man vide om rapporterne og projektet før man går i gang med at læse?

Indholdsfortegnelse

[Titelblad i](#_Toc115421481)

[(Indeks) ii](#_Toc115421482)

[Læsevejledning ii](#_Toc115421483)

[Forord ii](#_Toc115421484)

[(Indledning) 1](#_Toc115421485)

[Intro til problemet og processen, i modsætning til forord som er intro til rapporten og processen bagved. 1](#_Toc115421486)

[Case beskrivelse 1](#_Toc115421487)

[Problemformulering 1](#_Toc115421488)

[(Afgrænsning) 1](#_Toc115421489)

[Projektplanlægning 1](#_Toc115421490)

[Estimeret tidsplan 1](#_Toc115421491)

[Arbejdsfordeling 1](#_Toc115421492)

[Metode- og teknologivalg 1](#_Toc115421493)

[Væsentlige elementer fra produktrapporten 2](#_Toc115421494)

[Realiseret tidsplan 2](#_Toc115421495)

[Konklusion 2](#_Toc115421496)

[Diskussion 2](#_Toc115421497)

[(Referencer) 3](#_Toc115421498)

[(Bilag) 4](#_Toc115421499)

# Case beskrivelse

## Problemformulering

# Noget af det sværeste ved at tabe sig, er at holde sig motiveret til at fortsætte, og at se at det man gør, faktisk gør en forskel for ens krop og helbred. Selvom projektet ikke kan vise dine procenter og fortælle dig, hvordan din levetid har forbedret sig, kan den give dig en oversigt over din vægt, uden at du selv skal holde styr på det…

Så, hvordan kan det gøres nemmere at holde styr på ens vægt, uden at bruge for meget tid på, at skrive det ned i en bog, og hvordan kan produktet hjælpe med motivationsboost?

## (Afgrænsning)

Beskriv hvad du vil sortere fra din problemformulering for at kunne gennemføre projektet

# Projektplanlægning

## Estimeret tidsplan

Se Bilag A: Tidsplan.

## Arbejdsfordeling

Som klassisk programmør bruger man mere tid på programmering end rapportskrivning. Alt hvad der foregik dokumentation, både tidsplan, proces- og produktrapport, blev pænt lagt til side, så der var mere programmering. Det er nemmere at skrive om produkterne før og efter påbegyndelse, og så løbende opdatere tidsplanen.

Tidsplanen i sig selv var svær at arbejde med. Når du laver en proces for et produkt, vil du gerne starte i dele og gør de dele færdige, før du stater på nye dele, og har flere kørende på samme tid. Desværre har vi fag kørende i løbet af ugen med forskellige lærere, og man vil selvfølgelig gerne arbejde på sin app, når der står App III på skemaet…

Selve processen gik fint. Jeg ramte selvfølgelig ikke plet ved mange af mine punkter – nogle under og nogle over. Jeg havde ikke forventet at bruge super lang tid på selve appopsætningen, eller have store problemer med kamera og QR-kode, men det fik jeg.

Da .NET MAUI har problemer med deres kamera/billede funktion, kunne jeg ikke bruge min originale ide med kamera til at skanne en QR-kode for at oprette forbindelse. Da der ikke er andet der giver mening i mit projekt, har jeg ikke noget telefon hardware inkluderet i mit projekt.

# Metode- og teknologivalg

**App**

Der er mange forskellige teknologier til at lave apps. Ved app-design vælger man som regel en teknologi, der understøtter cross-platform. Xamarin & den nye .NET MAUI er nogle gode bud på C# appudvikling, og evt. Flutter & React Native som klassiske bud på JavasScript-like syntaks.

Af de 4 nævne frameworks har jeg allerede prøvet at udvikle i både MAUI og React Native. Jeg har selv valgt at arbejde med den nye MAUI frem for React Native eller at lære de andre frameworks at kende. Jeg føler, at det er nemmere at sætte et MAUI projekt op, og få appen emuleret frem for et React Native projekt. Jeg løber en lille risiko ved at bruge det nye MAUI, da der er mangel på hjælpemidler, og meget af frameworket stadig kan være fyldt med fejl eller ikke udviklet færdigt.

**API**

I forhold til API, er der primært C#’s ASP.NET Web API, JavaScripts Node ved brug af express.js, Pythons Django og Cs NodeMCU via ESP8266WebServer. Eftersom jeg har arbejdet i alle 3, undtagen Django, har det været lidt svært at beslutte, hvilken teknologi der giver mest mening. Både for projektet, men også for struktur og opsætning – hvad kræver mest, og hvor giver det mening at bruge de forskellige teknologier?

Jeg har valgt at gå med C#’s ASP.NET Web API. Jeg har ikke super meget erfaring med ESP8266WebServer eller Django, så de kunne jeg nemt krydse af listen. Selvom NodeMCU er relativ simpel, mangler jeg en bedre struktur for mit projekt. Derfor ville det næste blive JavaScripts Node. Jeg har meget erfaring med JavaScript, og har enligt ikke noget imod opsætningen, syntaks eller struktur. Dog har jeg valgt C#’s ASP.NET Web API over JavaScripts Node, fordi meget af det kode der er i Node er allerede i ASP.NET Web API. Derudover er det ikke lang tid siden, jeg sidst arbejdede med en ASP.NET Web API ift. en Node API.

**Database**

Databaser er for det meste delt op i relationel/non-relationel. I princippet kunne jeg have brugt begge dele:

* Relationel: Fordi mine modeller følger en bestemt struktur, som ikke har nogle valgfrie properties udover mine vægtmålinger. Brugerne har relation til deres målinger i form af 1..\*. En bruger kan have mange målinger, men 1 måling har kun 1 bruger.
* Non-relationel: Fordi mine målinger gemmes i masse. I en non-relationel database, ville det betyde at jeg kan gruppere mit data via bruger id, og få json objekter tilbage, fremfor at gemme *alle* målinger i samme tabel.

**Embedded**

Af embeddede enheder kender jeg til ESP8266, ATMega168 & Raspberry Pi. Min embeddede enhed skal kunne læse vægten på et objekt og sende dataen til mit API. Derfor skal den have et komponent til at læse vægt, og kunne være i stand til at sende dataen videre via internetforbindelse.

Til projektet bruger jeg en ESP2866, fordi den har minimumskravene på produktet. Har jeg tid til det, vil jeg også bruge en Raspberry Pi, da vi skal bruge en Pi i Linux/Embedded faget.

# Væsentlige elementer fra produktrapporten

Eks. henvisning til afsnit og punkter i produktrapporten

Find relevante overskrifter baseret på indhold

# Proces

**Mindmap/Flowchart**

For at få de 4 store dele sat sammen, har jeg brug for at lave et mindmap/flowchart til mig selv, så jeg har ordentligt overblik over mit projekt. Her kan jeg dykke ned i hvordan jeg vil sætte mit projekt op og få det hele til at spille sammen ordentligt og optimalt. Det er også her, jeg finder ud af hvilke modeller/klasser, der skal bruges til projektet – både backend og frontend.

Se mit Original Flowchart

**API**

Som API teknologi besluttede jeg mig for at bruge ASP.NET Core. Jeg har tidligere lavet 3 andre API’er i ASP.NET Core, så derfor var det relativt nemt at gå til. API’et er det vigtigste led i et fullstack projekt, for uden API’et, er der ingen kommunikation mellem frontend, database og embeddede maskiner. Derfor synes jeg at det gav mening for mig at starte der.

Efter at have lavet modellerne i et library, tak til min flowchart proces, havde jeg også en god idé om, hvilke former for controllers, der skulle tilføjes til API’et.

Da jeg var færdig med de basale endpoints, kom jeg til redis-cache delen. Jeg skulle have en cache, der gemmer på den nuværende bruger, der bruger en bestemt vægt, og også de midlertidige målinger fra vægte, der ikke har en bruger tilknyttet. Jeg konkluderede så, at det var for meget arbejde ift. hvad det hjalp. Derfor bliver alt nu gemt i min SQL-database – både forbindelser mellem bruger og vægt og også vægt målinger, der ikke har en bruger tilknyttet. Dog hvis en vægt måling har været gemt i systemet i 10 minutter uden bruger, bliver den fortolket som ugyldig måling og automatisk slettet.

Tak til Swagger, som kommer med til test af endpoints i ASP.NET projekter, var det hurtigt og nemt at teste og fiske fejl, der var opstået under konstruktionen af API’et.

//TODO: Se API’ets endpoints eller læs om dem i produktrapporten.

**Database**

Som database bruger jeg Microsoft SQL. Jeg gemmer 4 tabeller, som er vist i bilaget her. Det hele er styret af EntityFramework fra mit library projekt. Hver kommunikation mellem backend og database foregår via min DbContext klasse, som er en ORM for mine modeller/klasser.

2 af mine tabeller, **Users** og **Weights**, er generel data omkring brugerne og vægtene. En bruger har flere målinger på evt. forskellige vægte, hvilke er der min **Measurements** tabel kommer i spil. Dog ved vægten ikke hvilken bruger, der bruger vægten, når der måles vægt. Uden bruger relation til min måling, er målingen ikke fuldendt. Derfor har jeg brug for en **Connections** tabel, som er et mellemled mellem bruger og vægt. Når der er en valid forbindelse mellem bruger og vægt, kæder API’et brugeren sammen med vægten, der er kommet ind fra ESP2866’eren.

//TODO: Se databasetabellerne.

# Realiseret tidsplan

Tidsplanen for mig har været svær at komme igennem. Da jeg har minimal erfaring med at sætte tidspunkt på mine udviklingsevner, har jeg fuldstændig misset mine skud og oveni lagt tidsplanen på hylden. Nogle få gange har jeg fået den opdateret med mine realiserede tidspunkter.

Se min Tidsplan.

# Konklusion

Skal hænge sammen med case beskrivelsen og problemformuleringen.

## Diskussion

Diskutere forskellige side af løsningen, fordele/ulemper.

Hvis der var mere tid, hvor kunne produktet udvides eller forbedres.

Reflekter over projektet

Hvad har du lært?

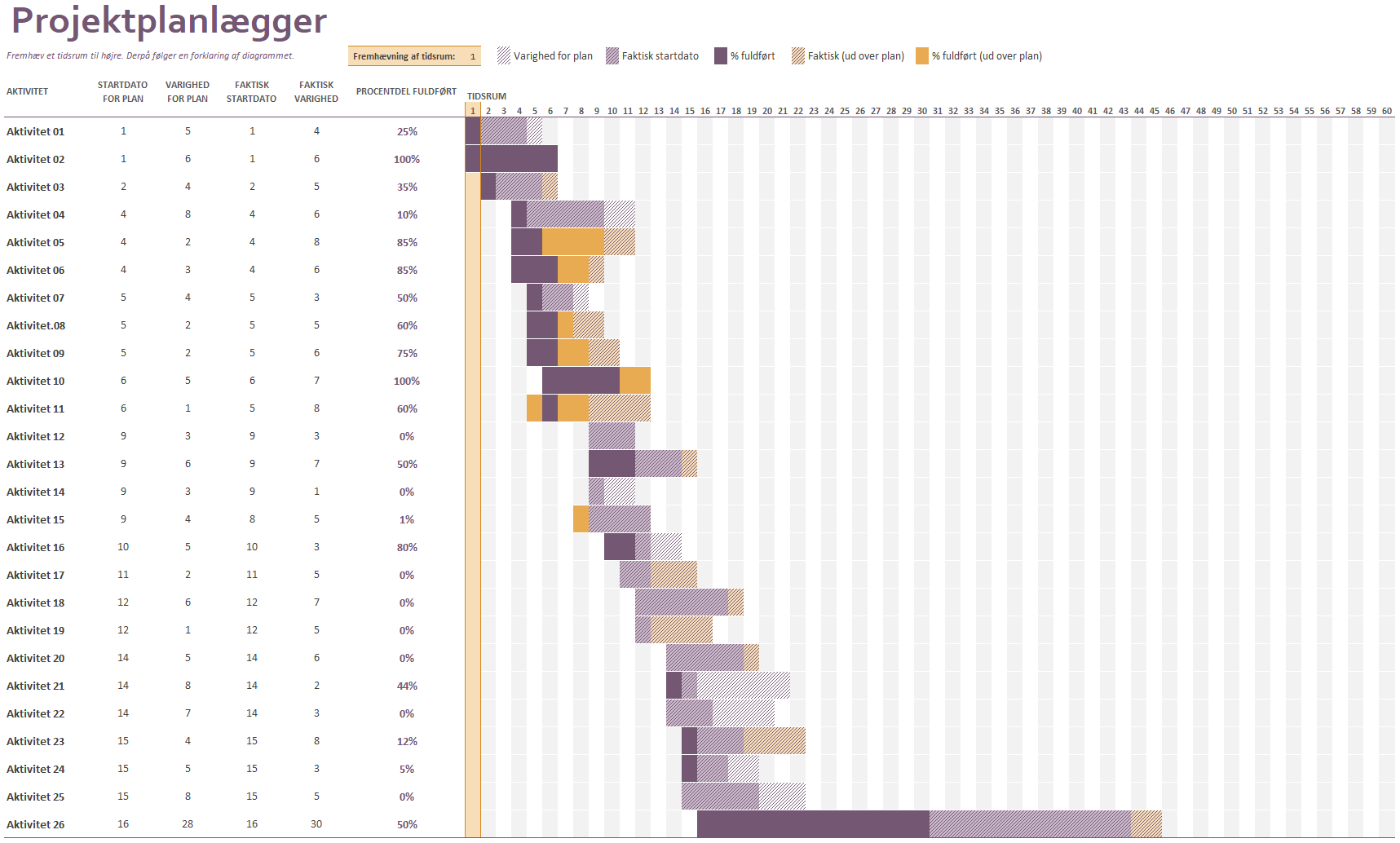
Valgte du de rigtige teknologier?

# Bilag

[Figur 1 placeholder billede 2](#_Toc115421472)

[Figur 2 Biag A: Estimeret Tidsplan 5](file:///C:\Users\dani146d\Documents\GitHub\Techcollege\Education\Hovedforløb%205\Tværfagligt%20Projekt\References\Processraport.docx#_Toc115421473)

#### Tidsplan



#### Original Flowchart

Diagram

Description automatically generated